EUROPEAN COMMISSION
DIRECTORATE-GENERAL
INFORMATICS
Information systems Directorate

European Commission

<Project Name> Test Management Plan

Date:	23/10/2008
Version:	1.002
Authors:	
Revised by:	
Approved by:	
Public:	
Reference Number:	

Commission européenne, B-1049 Bruxelles / Europese Commissie, B-1049 Brussel - Belgium. Telephone: (32-2) 299 11 11.

Commission européenne, L-2920 Luxembourg. Telephone: (352) 43 01-1.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1. Purpose 1

1.2. Scope 1

1.3. Intended Audience 2

1.4. Document Terminology and Acronyms 2

1.5. References 2

2. TARGET TEST ITEMS 2

3. OVERVIEW OF PLANNED TESTS 3

3.1. Overview of Test Inclusions 7

3.2. Overview of Other Candidates for Potential Inclusion 8

3.3. Overview of Test Exclusions 8

4. TEST STRATEGY 8

4.1. Measuring the Extent of Testing 8

4.2. Identifying and Justifying Tests 8

4.3. Conducting Tests 9

4.3.1. Functional Testing 9

4.3.2. Security Testing 11

4.3.3. Implementation Testing 12

4.3.4. Recovery Testing 13

4.3.5. User Interface Testing 15

4.3.6. Performance Testing 16

4.3.7. Load Testing 18

4.3.8. Stress Testing 19

4.3.9. Volume Testing 20

4.3.10. Configuration Testing 21

4.3.11. Installation Testing 22

4.3.12. Database Integrity Testing 23

4.3.13. Business Cycle Testing 24

4.3.14. Regression Testing 25

5. ENTRY AND EXIT CRITERIA 26

5.1. Project/ Phase Test Management Plan 26

5.1.1. Test Management Plan Entry Criteria 26

5.1.2. Test Management Plan Exit Criteria 26

5.1.3. Suspension and Resumption Criteria 26

6. DELIVERABLES 26

6.1. Test Evaluation Summaries 26

6.2. Reporting on Test Coverage 26
6.3. Perceived Quality Reports 26
6.4. Incident Logs and Change Requests 26
6.5. Regression Test Suite and Supporting Test Scripts 27
6.6. Traceability Matrices 27
6.7. Security Test Report 27
6.8. Additional Work Products 27
6.8.1. Detailed Test Results 27
6.8.2. Additional Automated Functional Test Scripts 27
6.8.3. Test Guidelines 27
7. TESTING WORKFLOW 27
8. ENVIRONMENTAL NEEDS 28
8.1. Base System Hardware 28
8.2. Base Software Elements in the Test Environment 29
8.3. Productivity and Support Tools 29
8.4. Test Environment Configurations 29
9. RESPONSIBILITIES, STAFFING, AND TRAINING NEEDS 30
10. KEY PROJECT/ PHASE MILESTONES 30
11. MASTER PLAN RISKS, DEPENDENCIES, ASSUMPTIONS AND CONSTRAINTS 30
12. MANAGEMENT PROCESS AND PROCEDURES 31
12.1. Managing Test Cycles 31
12.2. Approval and Signoff 31

Document History

Version	Date	Comment	Modified Pages

[Note: The following template is provided for use with RUP@EC. Text enclosed in square brackets and displayed in blue italics (style=Info Blue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).

The present document is a high level plan (master document) that will describe all test effort aspects (What, How, When, Where, Who) for a specific project using the Standard Development Case.

The Test Management Plan (TMP) is the main document that specifies all common testing aspects for a particular information system project. For each test iteration a specific Test Iteration Plan (TIP) will be created. The Test Iteration Plan will describe the detailed test effort as well as deviations and additional information from the TMP which serves as the master plan for the test effort.]

1. INTRODUCTION

1.1. Purpose

The purpose of the Test Management Plan of the *[complete with the name of your project]* is to:

- Provide a central artefact to govern the planning and control of the test effort. It defines the general approach that will be employed to test the software and to evaluate the results of that testing, and is the top-level plan that will be used by managers to govern and direct the detailed testing work.
- Provide all the necessary information for stakeholders interested in the testing discipline so as to (a) ensure that the testing activity is subject to proper governance and planning, (b) can deliver the necessary results.
- Serve as a plan for testing, subject to approval and validation from the stakeholders.

This *Test Management Plan also* supports the following specific objectives:

[The following is a list of representative objectives that you could address at this point. You may delete not relevant objectives listed below and modify or add missing additional objectives.]

- Identify the items that should be tested for the concerned project (high level).
- Identify and describe the test strategy that will be used to cover the test requirements.
- Identify the required resources and provide a high level estimate of the test effort.
- List the deliverables that will be provided during the test campaigns.
- List the major test activities.
- *[ISSP] [For projects of type A, B, C and D, lists the planned activities and acceptance criteria for testing the security features of the delivered system]*

1.2. Scope

*[Defines the types of testing —such as Functionality, Usability, Reliability, Performance, and Supportability—and if necessary the levels of testing—for example, Integration or System—that will be addressed by this **Test Management Plan**. It is also important to provide a general indication of significant elements that will be **excluded** from scope, especially where the intended audience might otherwise reasonably assume the inclusion of those elements.*

***Note**: Be careful to avoid repeating detail here that you will define in sections 2, Target Test Item, and 3, Overview of Planned Tests.]*

[Add specific information, delete not relevant items, complete missing items and modify existing text if necessary.]

This Test Management Plan applies to Integration, System and Acceptance tests that will be conducted on *[complete with the name of your application]*.It applies to test all requirements of the *[complete with the name of your application]* as defined in the Vision document, Use Case Specifications and the Supplementary Specifications.

Unit testing is considered as part of the development activities and it is assumed that unit testing has been successfully executed by the development team before proceeding to the tests specified in this and the TIP documents.

The Data Centre performance tests are not within the scope of this Test Management Plan, but the test scenarios and test cases for the performance tests will be supplied by the test team of the project.

The types of tests described in this document are based on the quality characteristics issued from ISO 9126 (Also known as FURPS+ in the RUP@EC terminology – FURPS+ is a system for classifying the requirements for the information system). The test types applicable for this project are described further in this document.

1.3. Intended Audience

*[Provide a brief description of the audience for whom you are writing the **Test Management Plan**. This helps readers of your document identify whether it is a document intended for their use, and helps prevent the document from being used inappropriately.*

***Note**: The document style and content usually alters in relation to the intended audience.*

This section should only be about three to five paragraphs in length.]

1.4. Document Terminology and Acronyms

*[This subsection provides the definitions of any terms, acronyms, and abbreviations required to properly interpret the **Test Management Plan**. Avoid listing items that are generally applicable to the project as a whole and that are already defined in the project's Glossary.*

A general Test Glossary containing all major and standard test terms, concepts and acronyms is defined for RUP@EC. You should refer to this Test Glossary but feel free to add specific test terms, concepts and acronyms specific to your test project in this section. Please avoid adding specific project related test terms, concepts and acronyms in the standard Test Glossary.]

All major test terms, test concepts and test acronyms are described in the Test Glossary document[1].

1.5. References

*[This subsection provides a list of the documents referenced elsewhere within the **Test Management Plan**. Identify each document by title, version (or report number if applicable), date, and publishing organisation or original author. Specify the sources from which the "official versions" of the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

2. TARGET TEST ITEMS

The list below identifies the test items—software, hardware, and supporting product elements —that have been identified as targets for testing.

[Provide a high level list of the major target test items. This list should include both items produced directly by the project development team, and items that those products rely on; for example, basic processor hardware, peripheral devices, operating systems, third-party products or components, and so forth. In the Test Management Plan, this may simply be a list of the categories or target areas.

[1] The Test Glossary document is located in the RUP@EC site Test Overview Page at http://www.cc.cec/CITnet/methodo/process/workflow/ovu_test.htm

In fact, you should provide a high level list of target test items (Hardware and software products related) as for example:

- *The test levels (Unit, Integration, System, Acceptance) which are in the target test items.*
- *Interactions/Integration with application(s) xyz.*
- *Multi platforms compliance …*
- *Outputs of the application on different printers*
- *All components produced by the development team.*
- *Application must run with Unix and Windows OS*
- *Connectivity with protocols, e.g. TCP/IP and X25*
- *Functionalities running with different internet browsers (Firefox, IE, Opera;…)*
- *…etc*

Please remember to list what is not in the scope of the target test items.]

3. Overview of Planned Tests

[This section provides a high-level overview of the testing that will be performed.

In this section you will list a high level overview of all types of test that will be included and excluded from the test effort. As possible, provide also a list of what will be tested for all types of test. The way that these tests will be performed (answering the question 'How are tests performed') must be described in Test Strategy section of the document.

Below you will find a standard structure that can be adapted depending on the test requirements and your own planned tests.]

The listing below identifies the high level items that have been identified as planned tests.

This list represents what will be tested; functional and non-functional test requirements.

[All planned test requirements that are included in your test effort will be added to section 3.1; potential planned tests in section 3.2 and test exclusions in section 3.3]

Functional Testing

[Function testing of the target-of-test should focus on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that is, verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analysing the output or results.

List the test requirements which are subject to functional testing.]

Security Testing

[Security and Access Control Testing focuses on two key areas of security:

Application-level security, including access to the Data or Business Functions

System-level Security, including logging into or remotely accessing to the system

Based on the security you want, application-level security ensures that actors are restricted to specific functions or use cases, or they are limited in the data that is available to them.

List the test requirements which are subject to security testing.]

Implementation Testing

[Implementation testing generally refers to the process of testing implementations of technology specifications. This process serves the dual purpose of verifying that the specification is implementable in practice, and that implementations conform to the specification. This process helps to improve the quality and interoperability of implementations.

List the test requirements which are subject to implementation testing.]

Recovery Testing

[Failover and recovery testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software, or network malfunctions with undue loss of data or data integrity.

For those systems that must be kept running, failover testing ensures that when a failover condition occurs, the alternate or backup systems properly "take over" for the failed system without any loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures, or invalid database pointers and keys. Recovery processes are invoked, and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

List the test requirements which are subject to recovery testing.]

Interface Testing

Test of the interface between systems. Example Web Services .....;

User Interface Testing

[User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the UI provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate, or industry, standards.

List the test requirements which are subject to user interface testing.]

*[**Difficulties with GUI Testing***

GUI testing itself, at a user-testing level, is not a difficult concept, but with larger and more complex GUI programs being written it becomes harder to test these GUI's [1,2]. Writing and maintaining hand-written GUI tests is very time consuming [1]. Automated testing of GUI's is even more complex. Some problems with automated GUI testing is the size and complexity of the GUI itself. There are many different states in a GUI and different arrangements of GUI actions can lead to different states.
White states that any automated GUI testing tool should include:
- record and playback of physical events in the GUI;
- screen image capture and comparison;
- shell scripts to control and execute test runs of the GUI.
The above description of an automated testing tool allows a user to interact with the GUI to write testing scripts that can be reused later. However, there are problems in the GUI testing tools themselves that need to be solved. The most important is the map of the GUI components and how the object are named and selected by the GUI testing tool. If a GUI testing tool relies on the location of a mouse to perform a certain event, any resizing or movement of the GUI window will cause errors when the test is replayed.]

Performance Testing

[Performance testing is conducted to evaluate the compliance of a system or software component with specified performance requirements, such as response times, transaction rates and resource utilisation.The list of the tests that could belong to a suite of performance tests is listed and explained here below:]

Benchmark tests

[A benchmark test compares the performance of new or unknown target-of test to a known reference standard such as existing software measurements. For example: PC magazine laboratories frequently test and compare several new computers or computer devices against the same set of application programs, user interactions, and contextual situations. The total context against which all products are measured and compared is referred to as the benchmark.

List the test requirements wich are subject to benchmark tests.]

Contention tests

[Verifies the target-of-test can acceptably handle multiple actor demands on the same resource (datarecords, memory, and so forth.

List the test requirements wich are subject to contention tests.]

Performance Profiling

[Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviours as a function of conditions, such as workload or hardware configurations. List the test requirements which are subject to performance testing.

List the test requirements wich are subject to performance profiling]

Load Testing

[Load testing is a performance test that subjects the target-of-test to varying workloads to measure and evaluate the performance behaviours and abilities of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time-sensitive issues.

List the test requirements which are subject to load testing.]

Stress Testing

[Stress testing is a type of performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources, like database locks or network bandwidth, although some of these tests are usually addressed under functional and load testing.

List the test requirements which are subject to stress testing.]

Volume Testing

[Volume testing subjects the target-of-test to large amounts of data to determine if limits are reached that causes the software to fail. Volume testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database, and would check that the software behaved normally and produced the correct report.

List the test requirements which are subject to volume testing.]

Endurance Testing

[The endurance testing is a load testing during a defined extended period of time in order to check the application and infrastructure stability (no memory leaks, availability of resources, etc) under load conditions.

List the test requirements which are subject to endurance testing.]

Bottleneck Detection

[The bottleneck detection is the process of finding the slowest part of the application using specialized introspection tools. Depending on the technology used to develop the application, the output of this phase can range from general information (ex: which tier is impacting the performance) to very detailed findings (ex: what SQL statement, what EJB is responsible).

List the test requirements which are subject to bottleneck detection.]

Configuration Testing

[Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections, and database servers vary. Client workstations may have different software loaded (for example, applications, drivers, and so on) and, at any one time, many different combinations may be active using different resources.

List the test requirements which are subject to configuration testing.]

Installation Testing

[Installation testing has two purposes. The first is to ensure that the software can be installed under different conditions (such as a new installation, an upgrade, and a complete or custom installation) under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify that, once installed, the software operates correctly. This usually means running a number of tests that were developed for Function Testing.

List the test requirements which are subject to installation testing.]

Database Integrity Testing

[The databases and the database processes should be tested as an independent subsystem. This testing should test the subsystems without the target-of-test's User Interface as the interface to the data.

List the test requirements which are subject to database integrity testing.]

Business Cycle Testing

[Business Cycle Testing should emulate the activities performed on the <Project Name> over time. A period should be identified, such as one year, and transactions and activities that would occur during a year's period should be executed. This includes all daily, weekly, and monthly cycles, and events that are date-sensitive, such as ticklers.

List the test requirements which are subject to business cycle testing.]

[OTHERS]

3.1. Overview of Test Inclusions

*[Provide a high-level overview of the major testing planned for project. Note what will be included in the plan and record what will explicitly **not** be included in the following section titled Overview of Test Exclusion.]*

3.2. Overview of Other Candidates for Potential Inclusion

[Give a separate overview of areas you suspect might be useful to investigate and evaluate, but that have not been sufficiently researched to know if they are important to pursue.]

3.3. Overview of Test Exclusions

[Provide a high-level overview of the potential tests that might have been conducted but that have been **explicitly excluded** *from this plan. If a type of test will not be implemented and executed, indicate this in a sentence stating the test will not be implemented or executed and stating the justification, such as:*

- *"These tests do not help achieve the evaluation mission."*
- *"There are insufficient resources to conduct these tests."*
- *"These tests are unnecessary due to the testing conducted by xxxx."*

As a heuristic, if you think it would be reasonable for one of your audience members to expect a certain aspect of testing to be included that you will not or cannot address, you should note its exclusion: If the team agrees the exclusion is obvious, you probably don't need to list it.]

4. TEST STRATEGY

[The Test Strategy presents an overview of the recommended strategy for analysing, designing, implementing and executing the required tests. Sections 2, Target Test Items, and 3, Overview of Planned Tests, identified ***what*** *items will be tested and* ***what*** *types of tests would be performed. This section describes* ***how*** *the tests will be realised.*

[ISSP] [For projects of type A, B, C and D, a strategy is defined for testing the Security features of the system.]

4.1. Measuring the Extent of Testing

[Describe what strategy you will use for measuring the progress of the testing effort. When deciding on a measurement strategy, it is important to consider the following advice from Cem Kaner, 2000 "Bug count metrics reflect only a small part of the work and progress of the testing group. Many alternatives look more closely at what has to be done and what has been done. These will often be more useful and less prone to side effects than bug count metrics."

A good measurement strategy will report on multiple dimensions. Consider the following dimensions, and select a subset that is appropriate for your project context: coverage (against the product and/ or against the plan), effort, results, obstacles, risks (in product quality and/ or testing quality) and historical trend (across iterations and/or across projects).]

Measurement of test progress is specified in the Measurement Plan.

[Explain any deviation from the Measurement Plan and document any additional measurements if necessary.]

4.2. Identifying and Justifying Tests

[Describe how tests will be identified and considered for inclusion in the scope of the test effort covered by this strategy. Provide a listing of resources that will be used to stimulate/ drive the identification and selection of specific tests to be conducted, such as Initial Test-Idea Catalogs, Requirements documents, User documentation and/ or Other Reference Sources.

Refer to http://www.cc.cec/CITnet/methodo/process/workflow/test/co_tstidsctlg.htm for Test-Ideas.

Refer any other document supporting the test strategy.]

4.3. Conducting Tests

[One of the main aspects of the test strategy is an explanation of how the testing will be conducted, covering the selection of quality-risk areas or test types that will be addressed and the associated techniques that will be used. You should provide an outline here of how testing will be conducted for each technique: how design, implementation and execution of the tests will be done, and the criterion for knowing that the technique is both useful and successful. For each technique, provide a description of the technique and define why it is an important part of the test strategy by briefly outlining how it helps achieve the Evaluation Mission(s).

The types of tests described in this document are based on the quality characteristics issued from ISO 9126 - also known as FURPS+ in the RUP@EC terminology.

FURPS+ is a mnemonic subset of ISO 9126 (software quality attributes) for classifying information system requirements. The test types will cover the expected quality characteristics of the system (refer to test requirements to define test types).

Refer to the following webpage for more information about FURPS+: http://www.cc.cec/CITnet/methodo/process/workflow/requirem/co_req.htm]

See also http://www.cc.cec/CITnet/methodo/process/workflow/test/co_keyme.htm to have more info about the key measures of a test.

[Please adapt the following standard test strategy to your own project.]

4.3.1. Functional Testing

Function testing of the target-of-test should focus on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that are verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analysing the output or results. The following table identifies an outline of the testing recommended for each application.

Test Objective(s):	Exercise target-of-test functionality. Ensure proper application navigation, data entry, processing, and retrieval to observe and log target behaviour.
Technique:	Exercise each use-case scenario's individual use-cases flows or functions and features, using valid and invalid data, to verify that: • The expected results occur when valid data is used in all test cases. • The appropriate error or warning messages are displayed when invalid data is used. • Each business rule is properly applied. • The appropriate information is retrieved, created, updated and deleted.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*

Test Objective(s):	Exercise target-of-test functionality. Ensure proper application navigation, data entry, processing, and retrieval to observe and log target behaviour.
Test Resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	• All planned tests have been executed. • All identified defects have been addressed.
Special Considerations:	Availability of test data and appropriate test environment.

4.3.2. Security Testing

Security and Access Control Testing focuses on two key areas of security:

- Application-level security, including access to the Data or Business Functions
- System-level Security, including logging into or remotely accessing to the system

Application security testing ensures that, based upon the desired security, users are restricted to specific functions or are limited in the data that is available to them.

System security ensures that only those users granted access to the system are able to access the application and only through the appropriate gateways.

Test Objective(s):	Application security: verify that user can access only those functions / data for which their user type is provided permissions. System security: verify that only those users with access to the system and application are permitted to access them.
Technique:	• Function / Data Security: Identify and list each user type and the functions / data each type has permissions for. • Create tests for each user type and verify permission by creating transactions specific to each user type. • Modify user type and re-run tests for same users. In each case verify those additional functions / data are correctly available or denied. • System Access (see special considerations below)
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	For each known user type the appropriate function / data are available and all transactions function as expected and run in prior Application Function tests.
Special Considerations:	Access to the system must be reviewed / discussed with the appropriate network or systems administrator. This testing may not be required as it maybe a function of network or systems administration.

4.3.3. Implementation Testing

Implementation testing generally refers to the process of testing implementations of technology specifications. This process serves the dual purpose of verifying that the specification is implementable in practice, and that implementations conform to the specification. This process helps to improve the quality and interoperability of implementations.

Test Objective(s):	Implementation testing ensures that specifications, standards, policies, conventions and regulations are respected.
Technique:	Verify that application is compliant with expectations.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	All planned tests have been executed.
Special Considerations:	None.

4.3.4. Recovery Testing

Failover / Recovery testing ensures that an application or entire system can successfully failover and recover from a variety of hardware, software, or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly "take over" for the failed system without loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions (or simulated conditions) such as device I/O failures or invalid database pointers / keys. Recovery processes are invoked and the application / system is monitored and / or inspected to verify proper application / system / and data recovery has been achieved.

Test Objective(s):	Verify that recovery processes (manual or automated) properly restore the database, applications, and system to a desired, known, state. The following types of conditions are to be included in the testing: • Power interruption to the client. • Power interruption to the server. • Communication interruption via network server(s). • Interruption, communication, or power loss to DASD (Direct Access Storage Device) and or Raid controller(s). • Incomplete cycles (data filter processes interrupted, data synchronisation processes interrupted). • Invalid database pointer / keys. • Invalid / corrupted data element in database.
Technique:	Tests created for Application Function and Business Cycle testing should be used to create a series of transactions. Once the desired starting test point is reached, the following actions should be performed (or simulated) individually: • Power interruption to the client: power the PC down. • Power interruption to the server: simulate or initiate power down procedures for the server. • Interruption via network servers: simulate or initiate communication loss with the network (by physically disconnecting communication wires or power down network server(s) / routers). • Interruption, communication, or power loss to DASD (Direct Access Storage Device) and or Raid controller(s): simulate or physically eliminate communication with one or more DASD controllers or devices. Once the above conditions / simulated conditions are achieved, additional transactions should executed and upon reaching this second test point state, recovery procedures should be invoked. Testing for incomplete cycles utilises the same technique as described above except that the database processes themselves should be aborted or prematurely terminated. Testing for the following conditions requires that a known database state be

	achieved. Several database fields, pointers and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	In all cases above, the application, database, and system should, upon completion of recovery procedures, return to a known, desirable state. This state includes data corruption limited to the known corrupted fields, pointers / keys, and reports indicating the processes or transactions that were not completed due to interruptions.
Special Considerations:	• Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required. • Resources from the Systems (or Computer Operations), Database, and Networking groups are required. • These tests should be run after hours or on an isolated machine(s).

4.3.5. User Interface Testing

User Interface testing verifies a user's interaction with the software. The goal of UI Testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the applications. In addition, UI Testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Test Objective(s):	Verify the following: • Navigation through the application properly reflects business functions and requirements, including window to window, field to field, and use of access methods (tab keys, mouse movements, and accelerator keys). • Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.
Technique:	Create / modify tests for each window to verify proper navigation and object states for each application window and objects.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	Each window successfully verified to remain consistent with benchmark version or within acceptable standard.
Special Considerations:	Not all properties for custom and third party objects can be accessed.

4.3.6. Performance Testing

Performance testing measures response times, transaction rates, and other time sensitive requirements. The goal of Performance testing is to verify and validate the performance requirements have been achieved. Performance testing is usually executed several times, each using a different "background load" on the system. The initial test should be performed with a "nominal" load, similar to the normal load experienced (or anticipated) on the target system. A second performance test is run using a peak load.

Additionally, Performance tests can be used to profile and tune a system's performance as a function of conditions such as workload or hardware configurations.

Test Objective(s):	Validate System Response time for designated transactions or business functions under a the following two conditions: • Normal anticipated volume. • Anticipated worse case volume.
Technique:	• Use Test Scripts developed for Business Model Testing (System Testing). • Modify data files (to increase the number of transactions) or modify scripts to increase the number of iterations each transaction occurs. • Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see special considerations below).
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	• Single Transaction / single user: Successful completion of the test scripts without any failures and within the expected / required time allocation (per transaction). • Multiple transactions / multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.

Special Considerations:	• Comprehensive performance testing includes having a "background" load on the server. There are several methods that can be used to perform this, including:
	  • "Drive transactions" directly to the server, usually in the form of SQL calls.
	  • Create "virtual" user load to simulate many (usually several hundred) clients. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with "traffic."
	  • Use multiple physical clients, each running test scripts to place a load on the system.

	• Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for Performance testing should be either actual size, or scaled equally.

4.3.7. Load Testing

Load testing measures subjects the system-under-test to varying workloads to evaluate the system's ability to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (response times, transaction rates, and other time sensitive issues).

Test Objective(s):	Verify System Response time for designated transactions or business cases under varying workload conditions.
Technique:	• Use tests developed for Business Cycle Testing. • Modify data files (to increase the number of transactions) or the tests to increase the number of times each transaction occurs.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	Multiple transactions / multiple users: Successful completion of the tests without any failures and within acceptable time allocation.
Special Considerations:	• Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for load testing should be either actual size, or scaled equally.

4.3.8. Stress Testing

Stress testing is intended to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the software that aren't apparent under normal conditions. Other defects might results from competition for shared resource like database locks or network bandwidth. Stress testing identifies the peak load the system can handle.

Test Objective(s):	Verify that the system and software function properly and without error under the following stress conditions: • Little or no memory available on the server (RAM and Direct Access Storage Device). • Maximum (actual or physically capable) number of clients connected (or simulated). • Multiple users performing the same transactions against the same data / accounts. • Worst case transaction volume / mix (see performance testing above).
Technique:	• Use tests developed for Performance Testing. • To test limited resources, tests should be run on single machine, RAM and DASD on server should be reduced (or limited). • For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume / mix.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	All planned tests are executed and specified system limits are reached / exceeded without the software or software failing (or conditions under which system failure occurs is outside of the specified conditions).
Special Considerations:	Stressing the network may require network tools to load the network with messages / packets. The Direct Access Storage Device used for the system should temporarily be reduced to restrict the available space for the database to grow. Synchronisation of the simultaneous clients accessing of the same records / data accounts.

4.3.9. Volume Testing

Volume Testing subjects the software to large amounts of data to determine if limits are reached that cause the software to fail. Volume testing also identifies the continuous maximum load or volume the system can handle for a given period. For example, if the software is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

Test Objective(s):	Verify that the application / system successfully functions under the following high volume scenarios: • Maximum (actual or physically capable) number of clients connected (or simulated) all performing the same, worst case (performance) business function for an extended period. • Maximum database size has been reached (actual or scaled) and multiple queries / report transactions are executed simultaneously.
Technique:	• Use tests developed for Performance Testing. • Multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume / mix (see stress test above) for an extended period. • Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries / report transactions simultaneously for extended periods.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	All planned tests have been executed and specified system limits are reached / exceeded without the software or software failing.
Special Considerations:	What period of time would be considered an acceptable time for high volume conditions (as noted above)?

4.3.10. Configuration Testing

Configuration testing verifies operation of the software on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded (e.g. applications, drivers, etc.) and at any one time many different combinations may be active and using different resources.

Test Objective(s):	Validate and verify that the client Applications function properly on the prescribed client workstations.
Technique:	• Use Integration and System Test scripts. • Open / close various PC applications, either as part of the test or prior to the start of the test. • Execute selected transactions to simulate user activities into and out of various PC applications. • Repeat the above process, minimising the available conventional memory on the client.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	For each combination transactions are successfully completed without failure.
Special Considerations:	• What PC Applications are available, accessible on the clients? • What applications are typically used? • What data are the applications running (i.e. large spreadsheet opened in Excel, 100 page document in Word). • The entire systems, network servers, databases, etc. should also be documented as part of this test.

4.3.11. Installation Testing

Installation testing has two purposes. The first is to insure that the software can be installed on all possible configurations, such as a new installation, an upgrade, and a complete installation or custom installation, and under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, etc. The second purpose is to verify that, once installed, the software operates correctly. This usually means running a number of tests that were developed for Function testing.

Test Objective(s):	Verify and validate that the client software properly installs onto each client under the following conditions: • New Installation, a new machine, never installed. • Update machine previously installed with same version. • Update machine previously installed with older version.
Technique:	• Manually or develop automated scripts to validate the condition of the target machine (new - never installed, same version or older version already installed). • Launch / perform installation. • Using a predetermined sub-set of Integration or System test scripts, run the transactions.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	Transactions execute successfully without failure.
Special Considerations:	What transactions should be selected to comprise a confidence test that the application has been successfully installed and no major software components are missing?

4.3.12. Database Integrity Testing

The databases and the database processes should be tested as separate systems. These systems should be tested without the applications (as the interface to the data). Additional research into the DBMS needs to be performed to identify the tools / techniques that may exist to support the testing identified below.

Test Objective(s):	Ensure Database access methods and processes function properly and without data corruption.
Technique:	• Invoke each database access method and process, seeding each with valid and invalid data (or requests for data). • Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved (for the correct reasons)
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	All database access methods and processes function as designed and without any data corruption.
Special Considerations:	• Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases. • Processes should be invoked manually. • Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events.

4.3.13. Business Cycle Testing

Business Cycle Testing should emulate the activities performed on the system over time. A period should be identified, such as one year, and transactions and activities that would occur during a year's period should be executed. This includes all daily, weekly, monthly cycles and events that are date sensitive, such as ticklers.

Test Objective(s):	Ensure proper application and background processes function according to required business models and schedules.

Technique:	• Testing will simulate several business cycles by performing the following:
	  • The tests used for application function testing will be modified / enhanced to increase the number of times each function is executed to simulate several different users over a specified period.
	  • All time or date sensitive functions will be executed using valid and invalid dates or time periods.
	  • All functions that occur on a periodic schedule will be executed / launched at the appropriate time.
	  • Testing will include using valid and invalid data, to verify the following:
	  • The expected results occur when valid data is used.
	  • The appropriate error / warning messages are displayed when invalid data is used.
	  • Each business rule is properly applied.

Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	• All planned tests have been executed. • All identified defects have been addressed.
Special Considerations:	• System dates and events may require special support activities. • Business model is required to identify appropriate test requirements and procedures.

4.3.14. Regression Testing

Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

Test Objective(s):	Verify that all functions work properly after code changes in new builds/releases.
Technique:	Run all test cases of the previous build/iteration/release. There is no formal regression testing level (stage) but regression testing is conducted as needed.
Test Oracles:	*[A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.* *Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made, and the characteristics of specific outcome that indicate probable success or failure.]*
Test resources:	*[Any document and/or tools used to test.]*
Completion Criteria:	All planned tests have been successfully executed.
Special Considerations:	None

[Other test techniques. It could be Semantic Testing, Syntactic Testing,...]

5. Entry and Exit Criteria

5.1. Project/ Phase Test Management Plan

5.1.1. Test Management Plan Entry Criteria

*[Specify the criteria that will be used to determine whether the execution of the **Test Management Plan** can begin.]*

5.1.2. Test Management Plan Exit Criteria

*[Specify the criteria that will be used to determine whether the execution of the **Test Management Plan** is complete or that continued execution provides no further benefit.]*

5.1.3. Suspension and Resumption Criteria

[Specify the criteria that will be used to determine whether testing should be prematurely suspended or ended before the plan has been completely executed, and under what criteria testing can be resumed.]

6. Deliverables

*[In this section, list the various artefacts that will be created by the test effort that are useful deliverables to the various stakeholders of the test effort**. Don't list all work products; only list those that give direct, tangible benefit to a stakeholder** and those by which you want the success of the test effort to be measured.]*

6.1. Test Evaluation Summaries

[Provide a brief outline of both the form and content of the test evaluation summaries, and indicate how frequently they will be produced.]

The Test Evaluation Summary is a formal artefact that contains the test results as well as test coverage information.

A Test Evaluation Summary will be produced: *[Indicate the frequency it will be produce and add more detailed information about this document if appropriate and/or necessary.]*

6.2. Reporting on Test Coverage

All test measurements are defined and described in the Measurement Plan. Part of test coverage information will be produced in the Test Evaluation Summaries.

For all other test measurements, the Software Development Plan will define which ones will be used.

6.3. Perceived Quality Reports

All test measurements are defined and described in the Measurement Plan.

For all other test measurements, the Software Development Plan will define which ones will be used.

6.4. Incident Logs and Change Requests

[Provide a brief outline of both the method and tools used to record, track, and manage test incidents, associated change requests, and their status.]

6.5. Regression Test Suite and Supporting Test Scripts

[Provide a brief outline of the test assets that will be delivered to allow ongoing regression testing of subsequent product builds to help detect regressions in the product quality.]

6.6. Traceability Matrices

[Using a tool such as Rational RequisitePro or MS Excel, provide one or more matrices of traceability relationships between traced items.

[ISSP][For projects of type A, B, C and D, take into account the Security requirements of the system.]

6.7. Security Test Report

[ISSP][For projects of type A, B, C and D, this section must contain the purpose/objectives of the test, the methodology used, major findings and conclusions, recommendations, initiatives and actions planned and a description of the benefits uncovered as a result of the tests.

6.8. Additional Work Products

*[In this section, identify the work products that are optional deliverables or those that should not be used to measure or assess the successful execution of the **Test Management Plan**.*

These Additional work products are optional and depend on stakeholder and project management needs.

These work products can be employed to improve the testing process at a next iteration and serve towards a continuous improvement of test effort and quality of the product.]

6.8.1. Detailed Test Results

[This denotes either a collection of Microsoft Excel spreadsheets listing the results determined for each test case, or the repository of both test logs and determined results maintained by a specialised test product.]

6.8.2. Additional Automated Functional Test Scripts

[These will be either a collection of the source code files for automated test scripts, or the repository of both source code and compiled executables for test scripts maintained by the test automation product.]

6.8.3. Test Guidelines

[Test Guidelines cover a broad set of categories, including Test-Idea catalogs (refer to http://www.cc.cec/CITnet/methodo/process/workflow/test/co_tstidsctlg.htm), Good Practice Guidance, Test patterns, Fault and Failure Models, Automation Design Standards, and so forth.]

7. TESTING WORKFLOW

*[Provide an outline of the workflow to be followed by the Test team in the development and execution of this **Test Management Plan**.]*

The specific testing workflow should explain how the project has customised the base RUP test workflow (typically on a phase-by-phase basis). It might be both useful and sufficient to simply include a diagram or image depicting your test workflow.

More specific details of the individual testing tasks are defined in a number of different ways, depending on project culture; for example:

- defined as a list of tasks in this section of the Test Management Plan, or in an accompanying appendix

- defined in a central project schedule (often in a scheduling tool such as Microsoft Project)
- documented in individual, "dynamic" to-do lists for each team member, which are usually too detailed to be placed in the Test Management Plan
- documented on a centrally located whiteboard and updated dynamically
- not formally documented at all

Based on your project culture, you should either list your specific testing tasks here or provide some descriptive text explaining the process your team uses to handle detailed task planning and provide a reference to where the details are stored, if appropriate.

For Test Management Plans, we recommend avoiding detailed task planning. This is often an unproductive effort, if done as a front-loaded activity at the beginning of the project. The planning aspects are part of the Software Development Plan (SDP) and should be documented in the SDP. Nevertheless, it may be useful to describe the phases and the number of iterations in the Test Management Plan. In such a case you may also give an indication of what types of testing are generally planned for each Phase or Iteration.

***Note**: Where process and detailed planning information is recorded centrally and separately from this Test Management Plan, you will have to manage the issues that will arise from having duplicate copies of the same information. To avoid team members referencing out-of-date information, we suggest that in this situation you place the minimum amount of process and planning information within the Test Management Plan to make ongoing maintenance easier and simply reference the "Master" source material.]*

[Basically, standard test tasks (to be considered in planning) are the following:

- *Plan Test*
- *Design Test*
- *Implement Test*
- *Execute Test*
- *Evaluate Test]*

8. Environmental Needs

*[This section presents the non-human resources required for the **Test Management Plan**.]*

8.1. Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Test Management Plan*.

[The specific elements of the test system may not be fully understood in early iterations, so expect this section to be completed over time. We recommend that the system simulates the production environment, scaling down the concurrent access and database size, and so forth, if and where appropriate.]

System Resources		
Resource	**Quantity**	**Name and Type**
Pentium IV with LAN connection	*3*	*Xxx*
Pentium III with LAN connection	*1*	*Xxx*

System Resources		
Resource	**Quantity**	**Name and Type**
1 remote PC with internet connection		*Xxx*
Unix	*1*	*Xxx*
Sun Solaris	*1*	Xxx

8.2. Base Software Elements in the Test Environment

The following base software elements are required in the test environment for this *Test Management Plan*.

[The software element names/versions/type & other notes, in the table below, are indicative. Please define them as appropriate for your project.]

Software Element Name	Version	Type and Other Notes
Standard installation test profile		
Toad	*Xxx*	*Xxx*
Capture IT	*Xxx*	*Xxx*
Microsoft Office	*2003*	*xxx*
Internet Explorer	*6*	*Xxx*
Firefox	*2.0.0.6*	*Xxx*

8.3. Productivity and Support Tools

The following tools will be employed to support the test process for this *Test Management Plan*.

[The tools/names/vendors/versions, in the table below, are indicative. Please define them as appropriate for your project.]

Tool Category or Type	Tool Brand Name	Vendor or In-house	Version
Test repository	*ClearCase*	*IBM*	*6*
Test Manager	*TestManager*	*IBM*	*6*

8.4. Test Environment Configurations

The following Test Environment Configurations needs to be provided and supported for this project.

[The names/descriptions/physical configurations, in the table below, are indicative. Please define them as appropriate for your project.]

[Note: Don't forget to take into account the Mirella Hosting Guidelines.]

Configuration Name	Description	Implemented in Physical Configuration
Integration Test Environment	*Xxx*	*Xxx*
End-to-End Test Environment	*Xxx*	*Xxx*

Configuration Name	Description	Implemented in Physical Configuration
Production like environment	*xxx*	*xxx*
Standard profile ABC environment	*xxx*	*xxx*

9. Responsibilities, Staffing, and Training Needs

All roles and responsibilities are described in the Software Development Plan.

10. Key Project/ Phase Milestones

Planning aspects and project/phase milestones are described in the Software Development Plan.

11. Master Plan Risks, Dependencies, Assumptions and Constraints

[The risks related to the test effort should appear in the risk list for the project. It is recommended to avoid duplicating copies of the same kind of information. In this section, you can add a reference to the risk list of the project.

*List any dependencies identified during the development of this **Test Management Plan** that may affect its successful execution if those dependencies are not honoured. Typically these dependencies relate to activities on the critical path that are prerequisites or post-requisites to one or more preceding (or subsequent) activities You should consider responsibilities you are relying on other teams or staff members external to the test effort completing, timing and dependencies of other planned tasks, the reliance on certain work products being produced.]*

Dependency between	Potential Impact of Dependency	Owners

*[List any assumptions made during the development of this **Test Management Plan** that may affect its successful execution if those assumptions are proven incorrect. Assumptions might relate to work you assume other teams are doing, expectations that certain aspects of the product or environment are stable, and so forth.]*

Assumption to be proven	Impact of Assumption being incorrect	Owners

*[List any constraints placed on the test effort that have had a negative effect on the way in which this **Test Management Plan** has been approached.]*

Constraint on	Impact Constraint has on test effort	Owners

Constraint on	Impact Constraint has on test effort	Owners

12. Management Process and Procedures

Test management processes and procedures that will be used are defined in the Software Development Plan.

[Any deviation or additional information must be documented in this section.]

12.1. Managing Test Cycles

[Outline the management control process for a test cycle.]

12.2. Approval and Signoff

[Outline the approval process and list the job titles (and names of current incumbents) that initially must approve the plan, and sign off on the plans satisfactory execution.]